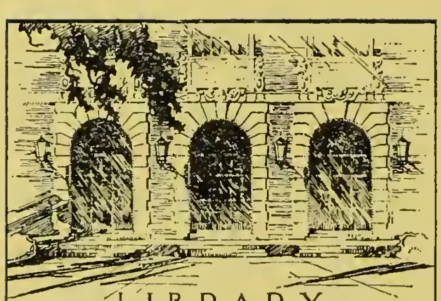
LIBRARY
OF THE
UNIVERSITY
OF ILLINOIS

510.84
Il6r
no.204-211
cop.2

[redacted]

The person charging this material is responsible for its return on or before the **Latest Date** stamped below.

**Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.**

University of Illinois Library

JUN 18 1978

[illegible]

APR 21 1978

APR 11 REC'D

MAR 11 1998

L161—O-1096

I26N Report No. 206 Math COO-1469-0024
no. 206
cop. 2

# AN ELEMENTARY DISCUSSION OF THE TIME SHARE CONCEPT

by

Lester M. Holland, Jr.

July 5, 1966

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

THE LIBRARY OF THE
OCT 7 1966
UNIVERSITY OF ILLINOIS

Digitized by the Internet Archive
in 2013

http://archive.org/details/elementarydiscus206holl

Report No. 206

# AN ELEMENTARY DISCUSSION OF THE TIME SHARE CONCEPT

by

Lester M. Holland, Jr.

July 5, 1966

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

## TABLE OF CONTENTS

## PREFACE

This paper is a brief, elementary description and explanation of the time share concept including the goals, successes, and shortcomings of experimental time share systems. The language of this report is necessarily limited to that which can be easily understood by the layman (the potential user of a time share system), students, programmers, engineers, and others who desire a basically non-technical, yet thorough, description of modern computer systems.

Although the author has drawn on personal experience gained in working with the CLIC project, this report is general in the sense that it is not limited to one system. Several systems are mentioned (and some in detail), and specific examples have been included where the author feels they are useful in conveying details of the time share concept.

Examples of the use of three systems are included in the Appendix. For additional or more detailed information the reader is referred to the Bibliography.

The author wishes to express his sincere thanks to Mr. E. B. Hassler who directed the work on the CLIC project and to Mr. C. E. Carter who helped the author gain an appreciation of the intricasies of the ILLIAC II computer.

LMH - May, 1966

## SYNOPSIS

The evolution of the time share computer system is causing a dichotomy between user and system programmers. This is the result of making computers more accessible and easier to use from the user's viewpoint at the cost of requiring a complicated and intricate supervisor/monitor program to control the operations of the system.

The increased speed of the new generation of computers has created problems of efficiency and user communication to the extent that a single user cannot possibly utilize the full speed of a central processor. To provide a practical solution to these problems, various time share systems have been proposed, and several operating systems have provided extensive information on system requirements versus user advantages.

The various requirements of time share systems including efficiency, communication, interaction, and supervisor programs are explained in terms of creating a utility computer system that can provide a computer service much as a telephone system provides a communication service. Two systems (Project MAC and the MULTICS system) are discussed in some detail to give an example of the evolution of time share systems in the span of only a few years.

# 1. INTRODUCTION TO THE TIME SHARE CONCEPT

The development and successful operation of time-shared computer systems have exploded the mystique of the computer as an all-powerful, highly mysterious tool whose workings are known only to the initiated few. The concept of time-shared operation is a dynamic and dramatic change in the organization and use of computers aimed at several important objectives.

First, the user is permitted to work "face-to-face" with the computer; when solving a thought problem he can use such a system just as he would use a power tool to aid in physical labor. Second, a shared system makes operation more economical timewise for both the user and the system. The central processing unit (CPU) spends a larger portion of its time doing useful computations while the user receives his answers and output information in much less time than previously required in batch processing systems. Finally the shared system makes it possible for many potential users with small problems and those who need the use of a large computer only occasionally to afford the cost associated with computer use.

## 1.1 Characteristics of an Efficient System

An efficient time-share system shifts the emphasis from the computer itself to the problems to be solved by the computer and restores control of the computer to its logical and ultimate place--the user. While tending to mean many things to many people, time share is a panacea for many ills. The success of operating systems is proof that this revolutionary concept has indeed solved more problems than it has created--and many problems have cropped up during the evolution of the various experiments.

A versatile time-share system allows a user to work directly with a large computer whether the user is in the room with the machine or a thousand miles away connected only by a regular telephone line. What's more, the user thinks that he has exclusive use of the machine even though many other users may share this same illusion. The time-share concept is not complicated--only the machinations that have led to it. If the user is to succeed in using the computer as easily as he would use the telephone, he shouldn't clutter his thinking with the machinations of computer technology any more than he would clutter his mind with the intricasies of a telephone central office.

## 1.2 Definition and Limitation of Terms

The previous generation of computers (those produced as recently as only a few years ago) handled all problems in a manner called batch processing. This meant that if a number of different problems were to be solved, they were lined up in order, entered into the computer one at a time, and the answers were produced by the computer, again in the same order as the programs were input. Often the CPU worked only 10 or 20 percent of the time, the rest of the time being used for input-output (I/O) operations which by their very nature (card reading, printing, magnetic tape recording) are much slower than the electronic operations of the CPU.

In an effort to save CPU time and increase system efficiency the method of multiprogramming was developed and utilized. A computer is multiprogrammed when several programs are stored in core memory at one time whether the programs are batch-loaded or time-shared. Only one program is running at a given time: if program A needs to process input from a tape unit, the CPU switches control to program B and program B is run until the input data is ready for program A. This makes possible a tremendous increase in efficiency because a larger percentage of CPU time is being spent on useful work.

In many instances (manned space flights, etc.) it is necessary to perform mathematical operations and make decisions while an event is taking place. To meet this need, the on-line, real-time system combines two types of activity. First, it receives information about a current activity as soon as it happens. Second, an answer to a continuing problem is available while the input data values are still usable. On-line operation may or may not be time shared since it would be possible for a computer to execute a single program in real time with direct data inputs.

The most recent development in the area of computer systems is the multiprocessing system. Multiprocessing is the simultaneous operation of two or more independent computers (central processors) executing more or less independent programs with access to each other's internal memories.

## 1.3 Requirements of Supervisor Programs

When any of the above systems are in use, a supervisor or monitor program is necessary to act as a traffic cop flashing "stop" and "go" signals to the various inputs. The supervisor must also handle the following tasks:

1) accept input continuously from the users' remote terminals,
2) parcel out the time that each program is allowed to have control of the central processor,
3) schedule the central processor time for immediate responses,
4) protect each program from an errant neighbor and from itself,
5) manage I/O queques and CPU interrupts,
6) transfer programs between primary and secondary storage, and
7) maintain, sort, merge, and "garbage collect" on data and program files.[1]

The requirements of the supervisor program are indeed complex, but this complexity is compounded by the fact that each of the above tasks must be handled in such a manner that its individual parts will not interfere with any other task in progress.

The supervisor program is extremely complicated and is usually the result of the work of many expert system programmers. The supervisor program must foresee every possible operating condition or error and be able to handle each of these without the intervention of a human operator. (Actually the supervisor must be made "operator proof" so that if the operator does make a mistake, like switching tape units at the wrong time or pushing the wrong button, the system will not destroy either itself or the private information files of the individual users.)

---

[1]Martin Greenberger, "The Two Sides of Time-Sharing," Datamation, Vol. 11, No. 11, November, 1965, p. 33.

# 2. IMPERFECTIONS OF COMPUTER SYSTEMS

## 2.1 Batch Processing System

Anyone who has used a closed shop (batch processing) computer system will welcome and appreciate the advantages of a time-share system. The era of punching cards, turning these in at a window, waiting (at least half a day) for printed output, and finding a simple keypunch spelling error caused the entire problem to be rejected will be simply an unpleasant memory of the past when an efficient time-share system is available to the user.

If the above closed shop system seems inefficient, consider the following situation existing only a few years ago in Texas:

> A daily bus service at 4 p.m. transports decks of cards (computer jobs) to College Station (Texas A and M), 100 miles from Houston (University of Houston). Decks are picked up at the bus station at 7 p.m., run on the computer, and printed output is returned on a bus arriving in Houston at 5 a.m. the next morning. Breakdowns which occurred occasionally were principally due to bus trouble or computer machine downtime. On the whole, however, the service is reliable.[2]

### 2.1.1 Problems of Efficiency

In recent years computers have grown bigger, more powerful, more complex, faster, more versatile, smarter, more dreadful, more noxious, more populous, and more necessary. The modern computer is so fast that letting one batch or remote user monopolize the machine is like giving over an entire electric system for the use of an individual. A single user cannot possibly make efficient use of the new generation of machines. How, then, can a computer system be used efficiently? The answer to this question can be explained only after another, equally important problem is faced.

### 2.1.2 Problems of Communication

The computer appears in the form of a superhumanly fast, accurate, methodical, tireless worker who is illiterate, perversely literal, devoid of initiative, and so highly salaried that one must strive continually to keep

[2]E. I. Organick, "Use of Computers in Engineering Education," College of Engineering and Computing Center, University of Houston, 1962, p. 14.

"him" from running out of work. Consider the binary language of computers. It would be difficult to design a language that is more difficult or unnatural for humans to learn to use. How, then, does a human user communicate with a machine?

## 2.2 Communication with Computers

The user communicates with a computer by the use of a language that is understandable to both the man and the machine. Actually, the language is usually one that is understood only by the user, but one that can be decoded by the computer. Considering the possible tasks a computer can be ordered to perform makes it the most fascinating machine that man now interacts with. The computer has thus struck like a silent atomic explosion, liberating terrific energy and setting many investigators off on a chain reaction of new intellectual adventures--namely, how to come to terms with the computer and make it a more natural ally.

### 2.2.1 The Interactive System

The creation and development of this "more natural ally" has led to interactive time-share systems--a possible, practical solution to both the man-machine communication problem and the efficiency problem.

#### 2.2.1.1 Reasons for Development

The motivation for an interactive computer system (one that works with the user in a conversational mode) is to obtain fast response for program generation, debugging, and execution. The user retains control of the computing process, can modify his program, change parameters, and correct errors as required.

The distinguishing characteristic of an interactive system is the man-machine interface. This is usually a keyboard plus a device for viewing what is typed (either a printer or CRT display) and the system I/O program. The requirements of the human user are the major design factors for an interactive system while the efficient use of a computer is determined by the human engineering techniques used in the system development.

### 2.2.1.2 Characteristics of System

An effective interactive system must possess several equally important attributes. It should feature a simple, short, nonambiguous control language. Simple in this context implies that certain assumptions will be made by the system. (If a user wants to restart a program which is in execution, it should not be necessary to first stop the program. This can easily be done by the system.)

The system should provide a fast response to user commands. Following a command (say, LOAD FILE ALPHA), the user is all ready to begin working with his information, and long delays at this point are extremely annoying. The computer's actions (as seen by the user) should be under his complete control. Thus the user should be able to specify completely his program flow. Finally, the necessary routine labor should be supplied by the system. This consists of numbering lines, using standard I/O data formats, and other standard conveniences.

### 2.2.2 Concept of Man-Machine System

If a system has the above mentioned attributes, the user can interact intimately with the computer to arrive at a problem solution with the system supplying much of the labor from preprogrammed routines. This type of system allocates tasks between assemblages of men and machines in such a cooperative relationship that the designer of such a system is obliged to view the user and the machine as a single (man-machine) system. Indeed, people--like machines--have certain kinds of inputs, outputs, coding, storage, information transmission, and stability. This notion of unity, however, or similarity between man and machine stirs wonder in the modern man's conception of himself especially when considering the computer sciences in which the human characteristics must be matched with those of the machine.

# 3. TIME-SHARE SYSTEMS

Now that the user can communicate effectively with a large machine, the question of efficient machine use can be considered in some detail. As already suggested, the concept of a shared system makes it possible for many users to have access to a computer system. As examples will illustrate, this concept makes it possible for the CPU to do productive work a large part of the time.

## 3.1 Project MAC

An early version of a time-share system was demonstrated in November, 1961, at MIT. This continuing experiment is known by the acronym of Project MAC which has a double meaning: Machine Aided Cognition and/or Multi-Access Computer. The operating system is indeed an example of both terms. The heart of Project MAC is an IBM 7094 computer and the supervisor program which handles all communication between users and the system, schedules, I/O queques, bookkeeping, and detailed accounting. This effort is a major program devoted to research efforts on advanced computer systems and the exploitation of these systems.

### 3.1.1 Utility Concept of System

The ultimate result, hopefully, will be an intimate collaboration between a human user and an efficiently operating computer in a real-time dialogue on the solution of a problem in which each of the two contributes his best capabilities. This is the beginning of the development and operation of a community utility that is capable of supplying computer power to each "customer" where, when, and in the amount needed. Analogous to an electrical distribution system, it could provide individuals, industry, and business with logical tools to aid in intellectual work, just as electrical tools aid in physical work.

### 3.1.2 User Reactions and Acceptance

Acceptance of the system was accompanied by the kind of impatience with failures and shortcomings that is characteristic of customers of a public utility. Capacities of the system are limited, and users are often unable to log in because the system is already loaded. The system may not be in operation just when one is planning to use it. Enthusiasm mixed with a great deal of frustration on the part of the user is the most common reaction. In other words, the system is far from being as reliable and dependable as one might expect a public utility to be.

Users find most helpful the fact that the use of the system literally places at their fingertips a great variety of services for writing, debugging, and compiling programs and facilities for working on problems in their own particular fields through the use of appropriate problem-oriented languages. The MAC system is rapidly expanding through the addition of new languages and other utility programming aids. The system is now scheduled for operation 24 hours a day, 7 days a week.

### 3.1.3 Value of System to User

The MAC system involves real-time communication with human users. Each user is concerned only with his program and need not worry about the operation of other users' programs or the operation of the system supervisor. Fail safe operation of the system assures that no user's program will be affected by any other user and that the user's files will be preserved (even in the event of total system failure). Programs are processed almost immediately so that the value of the computer to engineers, scientists, and students is increased, system efficiency is increased, and debugging time is decreased. The results of the MIT experiment indicate that it is no longer a question of the feasibility of a time-share system, but rather, a question of how useful a system can be produced.

### 3.1.4 User Response Time

The performance figure of greatest interest to the user (response time) is the time interval between the issue of a command and the completion on the part of the computer of the required task. Response time depends on the scheduling algorithm, the character of the command, and the number of commands issued by other users.

The scheduling algorithm operates as follows. Each user is assigned an initial priority which depends on the size of the program to be run. Highest priority is assigned to the smallest programs. The highest-priority programs are allowed up to a maximum of 4 seconds before being interrupted; lower-priority programs are allowed to run for longer intervals (multiples of 4 seconds). The lower the priority, the longer is the allowed interval. If a program is not completed within the allowed interval, it is transferred from core memory to drum memory and its priority is automatically reduced by one.

It should be noted that while a program is running, the supervisor must still handle the input from and output to all of the other operating consoles. This is handled by means of a demand interrupt system. When a remote user has sent or is ready to receive information, the CPU is interrupted. It notes this condition, processes the I/O request, and then returns to the running program.

### 3.1.5 System Hardware Failures

MAC has been plagued with a number of hardware failures. Hardware failures are, however, much harder to diagnose in a multiple-access computer because of the impossibility of reproducing the machine condition at the time of failure. Often it is extremely difficult to determine if malfunctioning is the result of a hardware failure, a system-programming error, or even the result of an error in the logical design of the machine.

## 3.2 Development of MULTICS System

The mistakes made and the experience gained from MAC have provided some of the impetus for the development of a newer, more powerful system.

### 3.2.1 Design of a Machine--GE 645

General Electric is presently completing design of a machine (GE 645) that will handle many of the system requirements automatically--what can be done by electronics, logic, and hardware will be processed automatically without the need for detailed programming (software). The supervisor required, however, will be extremely complicated, but will operate more efficiently.

### 3.2.2 Features of Central Processor

The GE 645 will feature multiprogramming, multiprocessing capabilities to provide interactive programming, static inquiries, simultaneous batch processing, remote batch processing, real-time control, and communication between terminals. This will provide many advantages for the user including on-line editing in a conversational manner, CRT terminals that can request data displays and manipulate the information, and the ability to use card input and receive card-to-printer output.

### 3.2.3 Software Aspects of Central Processor

The heart of this new system is the GE 645 computer and a monitor/ supervisor program called MULTICS (Multiplex Information and Computing Service) which is an operating system program developed by MIT, Bell Telephone Laboratories, and General Electric. The concept of the MULTICS-645/I operating supervisor has been refined through more than 500 man-years of intensive study by some of the world's most knowledgeable, most experienced computer users and programmers. The intricasies of the operating supervisor program and equipment should be of interest to the professional programmer and the electronics engineer, but not to management or the general user. (Only a brief sketch will be given in this paper.)

### 3.2.4 Hierarchy of Storage

The MULTICS system increases operating efficiency by providing a hierarchy of storage including core memory, drum, discs, and magnetic tape units. Placement of files in the hierarchy of memory (based on the frequency of use) is invisible to the user. He need only ask for his file by a code name and the operating system will know where to locate the information. Memory protection also exists on an invisible basis: a user can keep any of his files confidential without knowing how the computer's registers function to protect his information. The user may also specify conditional access to his files: files may be made available for reading only, execution only, or for full access. In addition, the user can specify different levels of accessibility for specific other users.

### 3.2.5 Method of Segmenting-Paging

Another useful concept of the system is the fact that there is no relationship between the size of a program and the core memory size. (A common complaint in the past has been, "The problem is too large for core memory--it just can't be done!") To read a book from cover to cover does not mean that all pages must be exposed to the eyes at one time. Likewise, large programs can be paged through core memory in a similar manner, and only the currently active pages need to be stored in core. MULTICS handles fetching additional pages as required. This method of segmenting-paging provides more effective use of core: core space and time are saved. The user is not even aware of where his program is placed in core memory--_if_ his program is in core.

### 3.2.6 Operational Utility Aspects

When this system is fully operational, it will indeed be a type of community utility available to serve the needs of hundreds of users simultaneously. The term "community" utility is somewhat misleading since a user will need to be connected with the computing center only by a standard telephone line and may actually be hundreds or thousands of miles away.

## 3.3 Planning a Time Share System

The problems of planning an efficient time share system are extremely detailed and are intensified by the greater complexity of real-time installations.

### 3.3.1 Information Handling Aspects

The first step in planning a system is to understand and define the necessary information handling requirements. Even defining the specifications of a large system is a tortuous process which may include traditional practices that have been handed down over the years with little or no reason for merit.

In the development of a system, methods and procedures often evolve through a series of stages as exceptions are introduced or grafted on, new regulations are forced on the users, and unexpected bugs appear.

### 3.3.2 Testing of Software

In most projects the greatest problems arise in the testing of the supervisor programs. Planning which neglects the problem of testing can lead to extreme difficulties.

#### 3.3.2.1 Correction of Errors

Tracing and uncovering errors is more complicated because a single program does not retain control of the CPU during the entire test. Only as many terminals are added to a system do the more exotic errors begin to occur. It should be noted that when an error is produced, the machine does not stop.

### 3.3.2.2 Detection of Errors

If the error is detected (and this is a lucky case), the error type and location are made available to the supervisor/monitor program and an error diagnostic program is executed. Some of the newer and larger installations with more than one central processor have the capability of switching suspected units off line so they may be tested and/or repaired while the rest of the system continues to operate at only a slightly reduced speed and efficiency.

# 4. THE ILLINOIS OPERATING SYSTEM

Research in the area of time-share systems is being conducted at the University of Illinois by the Department of Computer Science.

## 4.1 System Components and Organization

The final operating system will utilize the ILLIAC II computer (which was designed and built at the University) as the CPU, a PDP-7 computer as an interface between the CPU and up to 64 remote console units, various levels or storage (core, drum, disc, and magnetic tape), and a single level interrupt system. The planning of the 64-console system is nearing completion and the implementation of some segments is now in progress.

## 4.2 Software Available to Users

The system will feature several languages including an interactive FORTRAN-type language known as CLIC (Console Language Immediately Compilable). This language makes it possible for a novice user (or any user prone to the errors of programming) to work with the compiler in a conversational mode. The programming interface between the user and the CPU will exist between two I/O conversion routines which will handle all information exchanges. These routines will also supply the necessary code conversions, buffer packing, the flagging of special conditions and other console requirements.

### 4.2.1 Description of CLIC

CLIC itself is a compiler (language translator) in which the modification of a single statement requires the modification of only the machine language code produced by the statement in question. Such a change does not require a complete recompilation of the program.

The justification of this method of operation lies in the fact that most programs require many changes, but each change is usually minor. Thus, if the program were recompiled at each correction, enormous amounts of time would be required. The language is an editable compiler in the strictest sense and will run within the ILLIAC II operating system.

### 4.2.2 Limitation of User's Programs

Since the ILLIAC II computer has a limited core memory (8,192 words), the various phases of CLIC (the compile, load-link, and execute phases) must be linked by a short control routine. Reserving core space for the supervisor, necessary user tables, and I/O buffer areas takes about one-fourth of the total core memory. The user gets what is left--but in many cases, there will not be room for an extensive user program. The entire system will be dependent on the limited core size, and this will tend to limit the total operating efficiency that can be achieved.

### 4.2.3 Present Status of CLIC

At the present time the compiler and loader sections are operational, and successful programs have been produced, revised, and rerun from a remote console. It is anticipated that work will be resumed on CLIC as soon as the time-share system (operational supervisor program) on ILLIAC II is basically operational.

# 5. PLANS FOR THE FUTURE

## 5.1 Communication Aspects

A careful look at the experiments conducted to date and the mistakes made in developing the operational monitor programs provides a glimpse of what must be done in the future. Certainly communication in the strictest sense is of paramount importance to the user who is many miles distant from the computing center.

For example, consider the case in one experimental system where the computer operator received a message on the master console from a remote user. The message read: "I'm alone, please decipher the above error message." The error message, however, appeared only at the user's console. If the message could not be understood by the remote user, then the entire process of communication was broken. Indeed, a great deal of helplessness and frustration will exist when there are communication failures of this nature.

A user also finds it annoying to wait long periods when his console is inactive (for instance, when his program is in execution). To alleviate this worry, the computer should send a null character (a non-printing character) occasionally to the consoles of users in execution. The null character simply rattles the console and lets the user know that the computer is still working on his program. This placebo is an important feedback communication to the user. A transmission interval of 1 to 5 seconds is adequate to prevent the deadly lack of action that is most disturbing to the human user.

## 5.2 Systems of the Future

The situation today in computation is about what it was after the turn of the century in the distribution of electricity. Many advancements and achievements are to be expected within the next few years.

One system of the future proposes a money-key, an extension of the credit card. Under this concept, when a customer makes a retail purchase, he presents his money-card to a clerk. The clerk inserts the card into a local terminal and enters on a keyboard the price and other information. The terminal relays all this information to a central computer that has access to account information of the local bank. The computer deducts the amount of the purchase from the customer's account and adds it to the merchant's account. No money has changed hands; there is no billing, no accounting, and no deferred payments.

Such a system has been proposed by Sperry Utah, a division of the Sperry Rand Corporation.

If the inventory number of the purchased article is included in the input information, it would be a very simple matter to have this system place orders with a national warehouse for inventory replacements. Thus one system could handle not only sales but also the details of inventory for many retail stores.

A possible use of systems of interest to electrical engineers performing circuit design is foreseen by Professor Dertouzos of MIT.

> It is not difficult to imagine a not so distant time when a designer seated before a console attains, after some man-machine dialog, a circuit design he considers satisfactory. He then asks the computer to search component-manufacturer's tapes for components that will meet circuit and designer requirements. Having done so, he then commands numerically controlled tools to fabricate a prototype of the circuit, while all related financial transactions are completed automatically.[3]

## 5.3 Conclusions

The present trend of system development indicates the future will bring about a dichotomy among the people associated with computers. The group of users will find themselves working easily with a vast utility that offers them almost unlimited computing services. The group of computer experts (programmers, electronics engineers, and computer scientists) will find themselves the masters of this vast utility.

The dichotomy will cause a wide gap between the person who can understand computer programming (as seen from the user's viewpoint) and the person who can write the complex supervisor/monitor programs necessary to control and direct the operations of the system. It is with this latter group that the ultimate responsibility of success or failure will rest.

---

[3]Nilo Lindgren, "Human Factors in Engineering, Part II: Advanced Man-Machine Systems and Concepts," IEEE Spectrum, Vol. 3, No. 4, April, 1966, p. 66.

The following is a portion of a session with the QUICKTRAN system on August 24, 1965. The session was conducted with an IBM 1050 console and a data phone interconnection.

```
001    READY  ;USER(CC0000,3636)

              FULL QUICKTRAN SERVICE AVAILABLE
001    READY  ;
101. -READY        LOAD (LHLOOP)
104.1+ALTER        LIST
101. =        CF   PROGRAM LHLOOP
102. =        C    L HOLLAND, UNIV OF ILLINOIS, DEPT OF COMPUTER SCIENCE,
 7/13/65.
103. =             DO 10 I = 1, 10
104. =             DO 20 J = 1, 100
105. =        20   X = I + J
106. =        10   PRINT 30, X
107. =        30   FORMAT (10X, F20.1)
108. =             END
104.1+ALTER        ALTERX
109. +READY        START(0)
106. =0 30                       101.0
106. =0 30                       102.0
106. =0 30                       103.0
106. =0 30                       104.0
106. =0 30                       105.0
106. =0 30                       106.0
106. =0 30                       107.0
106. =0 30                       108.0
106. =0 30                       109.0
106. =0 30                       110.0
108. =HALT END STATEMENT ENCOUNTERED DURING EXECUTION
109. +READY        COMMAND
101. -READY   ;FINISH

001    READY
```

User input is underlined. This program had been stored within the QUICKTRAN system. This session simply asked for a listing of the program and execution of the program. Note the complexity of the various symbols (+READY, -READY, +ALTER, etc.) which are supposed to have some obvious meaning to the user.

The following is an example of the CLIC language. Again the console used was an IBM 1050 with a data phone interconnection to the ILLIAC II computer. User input is underlined.

```
READY
!CLIC
1.0         A = J ++ K * LONGNAME
NNNNN       ILLEGAL OPERATOR WILL BE IGNORED.  IT IS    +
NNNNN       NAME TOO LONG.  ASSUMED TO BE  LONGNA
2.0         DO 10, I = 1, 10
NNNNN       EXTRA COMMA IN 'DO' WILL BE REMOVED.
3.0        !CORRECT 2.0
2.0    5    DO 10 I = 1, 10
3.0         X = A**2 + B*I / (A + B)
NNNNN       MIXED MODE EXPRESSION WILL BE EVALUATED IN FLOATING POINT.
4.0         IF (X-A) 4,4,5
5.0    4    PRINT 6, A, B, X
6.0    10   CONTINUE
7.0        !CORRECT
2.1         IF (B) 10,11,10
0.5         B = 4.5E-3
0.6         J = 10
0.7         K = 7
0.8         LONGNA = 3
0.0        !RESUME
7.0        !GO
FFFFF       MISSING 'END' STATEMENT
FFFFF       MISSING STATEMENT NUMBER:
7.0         CORRECT 6.5
6.5    11   PRINT 6, A, B, X
7.0    6    FORMAT ( 3F15.6)
8.0         END
9.0        !GO
```

At this point the program would be linked and executed. Answers would be output to the user and the user would be free to revise and rerun his program at any time.

The following is an example of a session with the present ILLIAC II system. Note that the user language is quite simple. When the file "TEST" was listed by the system, there are errors evident in the text. This is the result of a demand interrupt program that is not able to give fast enough response to input each character as it was typed. (The final operating system will feature a PDP-7 computer to handle this problem.)

```
LOGIN
 HELLO, WHO ARE YOU...
ILLIAC
THAT'S NICE

DEFINE FILE

INSERT

     THIS IS AN EXAMPLE OF THE USE OF A TIME-SHARE SYSTEM.  THIS INFORMATION
IS BEING INPUT TO THE ILLIAC II SYSTEM BY USING A TELETYPE CONSOLE CONNECTED
TO THE COMPUTER BY MEANS OF A DATA TELEPHONE.


     THIS INFORMATION WILL BE SAVED IN A FILE NAMED 'TEST' AND WILL BE
LISTED LATER.  AT THAT TIME, OTHER INFORMATION COULD BE ADDED, THE FILE EDITED,
COPIED, OR SCRAPPED←←←ED.  FOR SPELLING OR TYPING ERRORS, THE BACK POINTING
ARROW WILL DELETE THE ERROR.  (SEE LISTING BELOW AND NOTE THE DIFFERENCE.)

.
SAVE TEST

TIPSY
   TIPSY ON THE AIR
1 TIPSY IS A PROGRAMMABLE CALCULATOR THAT USES THE COMPUTER.
2 THE USER IS ALLOWED UP TO FIVE LINES FOR HIS PROGRAM.
3 THE LANGUAGE USED IS VERY SIMPLE.
4 I WILL NOW CLEAR MY LINES BY RE-TYPING THEM WITH ONLY
5 CARRIAGE RETURNS.
1
2
3
4
5
```

```
1 FOR X = 1 TO 5, FOR Y = 10 TO 50 STEP 10,
2 A = X**2 + Y**2, B=X*Y-5,..

RUN
X  =            +1.
Y  =           +10.  A  =  +.1010000000E+03  B  =            +5.
Y  =           +20.  A  =  +.4010000000E+03  B  =           +15.
Y  =           +30.  A  =             +901.  B  =           +25.
Y  =           +40.  A  =  +.1601000000E+04  B  =           +35.
Y  =           +50.  A  =  +.2501000000E+04  B  =           +45.
X  =            +2.
Y  =           +10.  A  =  +.1040000000E+03  B  =           +15.
Y  =           +20.  A  =  +.4040000000E+03  B  =           +35.
Y  =           +30.  A  =             +904.  B  =           +55.
Y  =           +40.  A  =  +.1604000000E+04  B  =           +75.
Y  =           +50.  A  =  +.2504000000E+04  B  =           +95.
X  =            +3.
Y  =           +10.  A  =  +.1090000000E+03  B  =           +25.
Y  =           +20.  A  =  +.4090000000E+03  B  =           +55.
Y  =           +30.  A  =             +909.  B  =           +85.
Y  =           +40.  A  =  +.1609000000E+04  B  =          +115.
Y  =           +50.  A  =  +.2509000000E+04  B  =          +145.
X  =            +4.
Y  =           +10.  A  =  +.1160000000E+03  B  =           +35.
Y  =           +20.  A  =  +.4160000000E+03  B  =           +75.
Y  =           +30.  A  =             +916.  B  =          +115.
Y  =           +40.  A  =  +.1616000000E+04  B  =          +155.
Y  =           +50.  A  =  +.2516000000E+04  B  =          +195.
X  =            +5.
Y  =           +10.  A  =  +.1250000000E+03  B  =           +45.
Y  =           +20.  A  =  +.4250000000E+03  B  =           +95.
Y  =           +30.  A  =             +925.  B  =          +145.
Y  =           +40.  A  =  +.1625000000E+04  B  =          +195.
Y  =           +50.  A  =  +.2525000000E+04  B  =          +245.
```

```
MASTER
LOAD TEST
PRINT 50
     THIS IS AN EXAMPLD OF THE USE OF A TME-SHARESYSTEM.  THIS INFORMATION
IS BEING INPUT TO THE ILLIAC II SYSTEM BY USING A TELETYPE CONSOLE CONNECTED
TO THE COMPUTER BY MEANS OF A DATA TELEPHONE.

     THIS INFORMATION WILL BE SAVED IN A FILE NAMED 'TEST' AND WILL BE LISTED
LATER.  AT THPT TIME, OTHER INFORMATION COULD BE ADDED, THE FILE EDITED,
COPIED, OR SCRAPED.  FOR SELLING OR TYPING ERRORS, THE BACK POINTING AROW
WILL DELETE THE ERROR.  (SEE LISTING BELOW AND NOTE THE DIFFERENCE.)

END OF FILE
LOGOUT
ILLEGAL COMMAND
SCRAP
LOGOUT
 GOODBYE
```

[1] Crisman, P. A. (ed.), The Compatible Time-Sharing System: A Programmer's Guide, 2ed., The M.I.T. Press, 1965.

[2] Desmonde, William H., Computers and Their Uses, Prentice-Hall, Inc., 1964.

[3] Desmonde, William H., Real-Time Data Processing Systems: Introductory Concepts, Prentice-Hall, Inc., 1964.

[4] Fana, R. M., "The MAC System: The Computer Utility Approach," IEEE Spectrum, Vol. 2, No. 1, January 1965, pp.56-64.

[5] "GE 645--A New Concept in Computer Time Sharing," General Electric Co., Computer Equipment Department, Phoenix, Arizona, 1965.

[6] Greenberger, Martin, "The Two Sides of Time-Sharing," Datamation, Vol. 11, No. 11, November 1965, pp. 33-36.

[7] Haas, Melvin E., "An Interactive Time Share System for a Small Computer," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, Report No. 196, February 1966.

[8] Hassler, Edwin B., Jr., "Comments on QUICKTRAN," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, File No. 680-01, August 1965.

[9] Hassler, Edwin B., Jr. and T. A. Murrell, "Remote Consoles for ILLIAC II," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, File No. 605, July 1964.

[10] Holland, L. M., "CLIC Phases--Allocation of Memory," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, TSM-56-177, June 1965.

[11] Holland, L. M., "Input-Output Conventions: CLIC--1050," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, TSM-59-191, August 1965.

[12] Lindgren, Nilo, "Human Factors in Engineering, Part I: Man in the Man-Made Environment," IEEE Spectrum, Vol. 3, No. 3, March 1966, pp. 132-139.

[13] Lindgren, Nilo, "Human Factors in Engineering, Part II: Advanced Man-Machine Systems and Concepts," IEEE Spectrum, Vol. 3, No. 4, April 1966, pp. 62-72.

[14] Organick, E. I., "Use of Computers in Engineering Education," College of Engineering and Computing Center, University of Houston, 1962.

[15] "Quarterly Technical Progress Report," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, July-September, 1965.

[16] "Quarterly Technical Progress Report," Department of Computer Science, University of Illinois, Urbana, Illinois 61801, October-December, 1965.

[17] Riley, Wallace B., "Time-Sharing: One Machine Serving Many Masters," Electronics, Vol. 38, No. 24, November 29, 1965, pp. 72-78.

[18] "The Time-Shared Use of Computers," General Electric Co., Computer Equipment Department, Phoenix, Arizona, 1965.

OCT 1 7 1966

JUN 2 0 1969

UNIVERSITY OF ILLINOIS-URBANA
3 0112 039572844